TP1: Pokédex

Ana Gonçalves

Universidade Federal de Minas Gerais

I. Introdução

O presente trabalho tem como objetivo implementar um sistema modelo servidor e clientes para simular a interação de um treinador com sua Pokédex. O sistema deve possuir 4 mensagens principais, sendo elas: adicionar um novo Pokémon remover um Pokémon existente no servidor,listar a relação de Pokémons salvos no servidor e realizar uma troca entre dois Pokémons.
O código base que foi utilizado foi o disponibilizado na metaturma de Redes e explicado no vídeo Introdução à Programação em Redes.
As especificações do trabalho foram disponibilizadas na metaturma, assim como os testes.

II. Implementação

O trabalho foi feito utilizando o Linux, na sua distribuição Zorin OS 16, utilizando a linguagem C. Primeiro, como mencionado acima, ocorreu o acompanhamento das aulas da disciplina juntamente com o código fonte disponibilizado, a diferença é que para a pokédex a ser realizada não precisaria do "server-mt.c". O método "socket parser"permite a conexão do IPv4 e IPv6.
Cria-se uma matriz pokédex[40][11], devido a capacidade máxima da pokédex de 40 pokemons e do nome dele de até 10 caractéres (são 11 devido a possibilidade de existir um pokemon com 10 caracteres e o último caracter precisa ser '\0').
Cria-se uma função "clearPokedex()"de modo a inicializar e limpar a pokedex.
Cria-se diversas funções para que todas instruções pedidas sejam resolvidas.
Dentro do "main"chama-se a função "handle"que chamará as outras dependendo da mensagem pedida. Dentro dela há a validação da mensagem, já que devido a documentação os caracteres têm que ser letra minúscula ou número, após isso ele compara a mensagem recebida e vai para o comando indicado.
A função "handleAddPokemon()"tem que validar:
1 - Se a pokedex está cheia, foi criado uma função para essa verificação ("fullPokedex()"), que é chamada, caso esteja precisa dar a mensagem "limit exceeded". Caso não vai ao caso 2.
2 - Verificar se o pokemon já existe, também foi criado uma função para isso ("existsPokemon()") e, caso exista, a mensagem "pokemon already exists"com o nome do pokemon é enviada. Caso não vai ao caso 3.
3 - Após todas essas verificações caso nenhuma aconteça o pokemon é adicionado a pokedex, utilizando a função "findPlacePokedex()"que procura um lugar para ele ser adicionado e a mensagem enviada é "pokemon added".
OBS.: Existe a função "clearSpaceInFinal()"para quando foi enviar a mensagem não existir um espaço final, esse espaço final é reprovado quando é feito a correção semi-automática.
A função "handleRemovePokemon()"tem que validar:
1 - Se o pokemon existe, usa-se a função "existsPokemon()"caso ele não exista ele não pode ser deletado, assim imprime "pokemon does not exist"com o nome do pokemon. Caso não ocorra, vai ao caso 2
2 - No caso 2 remove o pokemon e imprime "pokemon removed"com o nome do pokemon.
A função "handleListPokemon()"tem que validar:
1 - Se a lista está vazia, caso esteja tem que imprimir "none". Caso não esteja vai imprimir todos os pokemons, um do lado do outro em ordem.
OBS.: O "clearSpaceInFinal()"é usado aqui pelo mesmo motivo citado anteriormente.
A função "handleExchangePokemon()"tem que validar:
1 - Se o pokemon que pede para ser trocado existe, caso não exista a mensagem "pokemon does not exist"com o nome do pokemon é enviada. Caso exista, vai ao passo 2.
2 - Se o pokemon que pede para ser adicionado já exista, caso já exista a mensagem "pokeon already exists"com o nome

do pokemon é enviada. Caso não exista, vai ao passo 3.
3 - O pokemon é trocado e a mensagem enviada é "pokemon exchanged"com o nome do pokemon.
OBS.: A função "formatString()"usada é feita para retirar qualquer '%n' que possa aparecer na string para que na hora das mensagens serem enviadas não haja formatação errada.
Todo o código do trabalho foi feito usando client.c, server.c, common.c, common.h e o Makefile.

III. Dificuldades

No geral, as maiores dificuldades encontradas no trabalho foram de formatação para que o envio das mensagens seja feito corretamente (o erro 1 dos testes automáticos), algumas funções foram implementadas para isso, como mencionado na parte de implementação acima.
Um erro encontrado foi o timeout nos testes automáticos (erro 7) que foi solucionado colocando '\n' ao final das mensagens.
Algumas dificuldades em relação a manipulação de matriz em C, assim como ponteiro, também foram resolvidas com a procura de alguns tutoriais no Google e ajuda de colegas, como por exemplo o tamanho da matriz ser [40][11], só foi notado a necessidade quando houve um teste com o charmander, que possui 10 caracteres.
A maior dificuldade foi em relação ao recebimento de mensagens particionadas em múltiplas partes, que foi resolvido com ajuda de colegas de classe que mencionara que era necessário que procurasse um '\n' enviado pelo cliente e que a partir dele ocorreria o tratamento.
Abaixo os testes automáticos no código rodados no meu computador.

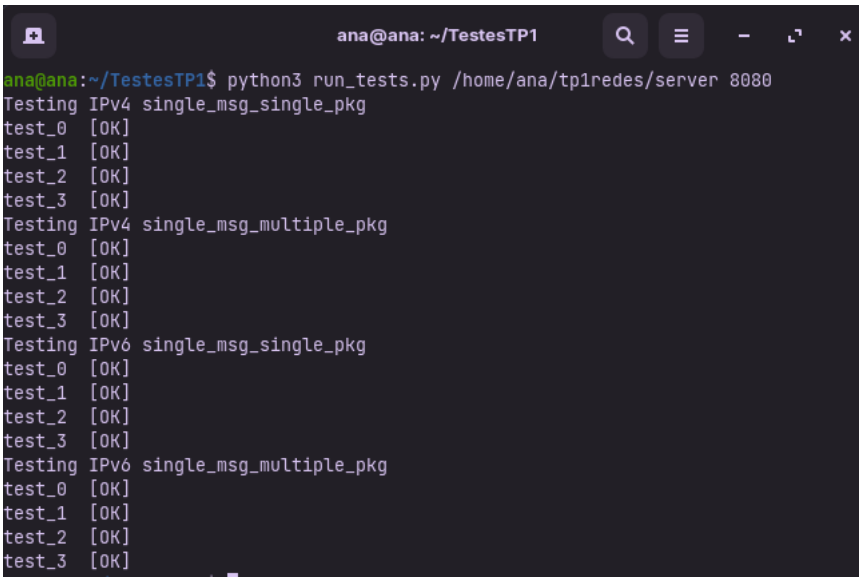

Figura 1: *Testes automáticos realizados*